University of Waterloo
Faculty of Engineering

Final Report

Project MUSE
(Music Usability Software Experiment)

Team Members:

Ram Sharma
Student ID: 20343914

Richard Lesco
Student ID: 20353896

Chris Vandevelde
Student ID: 20316110

Project Supervisor:
Dr. Hamid R. Tizhoosh
Systems Design Engineering Professor

Department of Systems Design Engineering
April 4, 2014

Executive Summary

A myriad of options currently exist for aspiring artists to produce music. However, upon attempts to use these options, users almost always encounter a combination of issues barring them from producing music. The instruments and tools (hardware and software) commonly used in music production have a steep learning curve, can be unintuitive, and are typically expensive. These obstacles can be difficult for a beginner to overcome. An experienced musician can happen upon inspiration for a musical idea at a time when they don't have the necessary tools at their disposal, rendering the idea inert. A solution is proposed involving a software platform to be run on readily available hardware (e.g. laptop, smartphone), which will translate user inputs (e.g. beat-boxing, drumming on a surface) into musical output via the industry standard MIDI (Musical Instrument Digital Interface) protocol. This solution effectively allows musicians to produce using the most intuitive tool available - their own bodies. Whereas visual artists and writers have access to significant rapid prototyping freedom using a pen and paper, available tools inhibit the creativity of musicians. The aim of Project MUSE is to facilitate rapid prototyping in music creation by providing a musical sketchpad for aspiring producers.

Various existing commercial products serve similar functions, evidencing the feasibility of the proposed solution. However, each of these solutions is mired with significant issues. They are either constrictively expensive and limit their use to specific environments, or they serve more as toys rather than professional products.

Academic research results indicate great potential for this project. A PhD thesis by Dan Stowell covers a very similar problem extensively, in which vocal inputs are parsed, classified and mapped to musical outputs. His deep research shows promising results, and his findings can be leveraged to assist the development of Project MUSE. Additionally, a significant amount of research has been performed in tangentially related fields such as speech recognition, wherein feature extraction and classification methods useful to this project have been explored extensively.

The system is prototyped in MATLAB, an environment known for its deep machine intelligence and signal processing capabilities as well as being conducive to rapid development. Usage is split into two distinct modes - one for percussive performances

(i.e. beatboxing) and another for melodic performances (i.e. singing or humming). The percussive mode requires classifier training in order to associate specific inputs with outputs. This process involves the user selecting desired output sounds, then for each distinct output providing an input via microphone audio. The melodic mode does not require classifier training, as it simply interprets note pitch and maps sounds to musical notes using a mathematical function. Both modes result in a MIDI file output, which can be used within any music production software to trigger a plethora of available virtual instruments.

The translation of an input performance to an output file involves two processes - note detection and note classification. Note boundary detection strategies were explored involving the comparison of input amplitude and frequency to a recorded noise floor, as well as N-Gram methods. Once notes are detected via their start and end times, they are extracted and classified using a Naive-Bayesian classifier, a method favoured for its simplicity and reliable accuracy. The classifier uses a variety of audio features such as amplitude, frequency, and length in order to match input notes to their previously correlated output sounds.

Project MUSE was tested using an automated suite. This suite provided performance metrics by running a number of preset input performances through the system and comparing the resulting outputs with preset desired outputs. The accuracy rating of the system is based on a weighted average of the error of note start and end times, as well as classification accuracy. The end result of accuracy testing yielded a rating of 89.20% for percussive performances, and a 39.78% rating for melodic performances.

Next steps for the project include expansion of the classifier and boundary detection methods, improvement of melodic performance interpretation, handling of overlapping notes, expanding to more programming languages and environments for the development of commercial products, incorporation of new input and output methods, improvements to the test suite, and real-time operation.

Overall, the project was a success in making the music production process significantly easier and has great future potential. Amateur and professional music production enthusiasts showed interest in using the project for their own purposes, evidencing its potential profitability as a commercial product.

Table of Contents

List of Figures

1.0 Introduction

1.1 Problem statement

Though many people in the world have the creative ability to make new music, a variety of obstacles exist which prevent the transformation of music from idea to audio. The potential artist may not know how to play all of the instruments involved in their idea, or they simply may not have access to those instruments. They may also be lacking the expensive hardware and software that is typically associated with music production. Even if these resources are available, they often have a steep learning curve and slow workflow or multiple people may be required to produce a single song. As the musician spends time learning to use these tools and going through the prescribed, and usually unintuitive, workflow the original musical idea gradually fades and is forgotten. Sadly, much potential music is lost while musicians attempt to navigate an expensive abyss of overwhelmingly complex interfaces. Even an experienced producer may happen upon an idea at an inopportune time away from their studio equipment, leaving the idea at the mercy of the human brain's fleeting short term memory. There currently does not exist a tool that allows musicians to sketch out their ideas, like a painter can with a sketchbook. Many tools have been created to help musicians sketch out their musical ideas, however most of them are usually gimmicky, end up being used as toys, and offer no value to the final version of a song. The other professional solutions lack flexibility in being able to produce different genres of music or are specific to proprietary software. A gap exists in the market for a tool that allows amateur musicians and music enthusiasts to produce high quality musical pieces in a mobile manner, while being as expressive as possible to account for all types of ideas and desires.

The goal of Project MUSE is to minimize the time and resources required for amateur musicians and music enthusiasts to bring a musical idea from idea to execution, creating the bulk of a complete musical piece using intuitive tools such as their voice, and readily available resources such as a smartphone or laptop. The time-consuming, resource-heavy, and complicated process of music production will be accelerated with a tool that has no need for an instruction manual. The potential users of this platform

include musicians in the early stages of their career, with limited resources, and music enthusiasts who enjoy experimenting with new technologies. The goal of Project MUSE includes translation of a user-generated input (e.g. beat-boxing sounds, drumming on a nearby surface, sung or hummed tones, accelerometer input, etc.) to an output consisting of real musical sounds (percussive and tonal) via use of the industry standard MIDI protocol. This will be achieved by a software platform employing pattern recognition and machine learning algorithms. The software will be able to run on readily available hardware such as a laptop or smartphone. The product will also compliment studio software and enable the user to easily integrate the tool within his/her existing tool-set.

1.2 Background information

State of the art

Software tools have proliferated with simulations of existing hardware such as samplers, keyboards and drum machines, as well as more generic interfaces involving the direct manipulation of notes in a variety of formats (including music notation and more simple abstract visualizations). For music production in a home setting, several categories of tools represent the options available to amateur musicians. Project MUSE aims to incorporate the beneficial features of each category of tool set in order to craft a robust solution aimed to satisfy all genres of music producers.

Keyboards and synthesizers

The keyboard presents a format which may be familiar to home musicians who have mastered the piano or any other instrument which uses a similar arrangement of keys. Synthesizers can output pure digital representations of the music being played and are often connected directly to computers to save the individual notes (as opposed to the audio output) which can later be modified or adjusted, even creating multiple final instruments from one original source.

The advantage of using a keyboard or synthesizer is that keyboards are ubiquitous in all genres of music and they allow artists to be expressive while exercising foundational knowledge of music theory. In addition, the keyboard layout has become an integral part of the MIDI protocol.

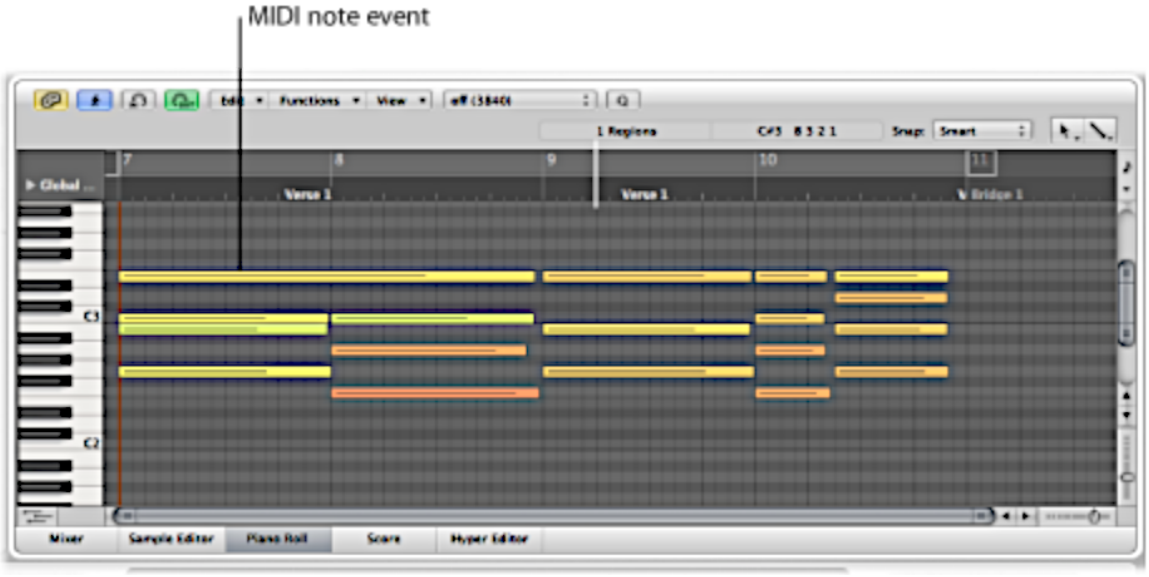

Figure 1 - MIDI Piano Layout

However, synthesizers and keyboards can be expensive and bulky. Project MUSE intends to emulate certain keyboard features such as velocity sensitivity and the piano roll layout available via MIDI. Velocity sensitivity is an important factor in a final piece of music because when playing an instrument all artists naturally don't have the same amplitude or loudness with each note. By incorporating velocity sensitivity into the final prototype, the final output can be made to sound as realistic as possible.

Abstract samplers

Samplers or drum machines such as the AKAI MPC2000 [1] or the Roland TR-808 [2] let users program sequences of samples (such as recorded drum sounds) and have it play them back in different arrangements, adjusting tempo and effects to produce a rhythmic track. In traditional abstract samplers, such as the MPC2000, each sample is associated with a pad, with 16 pads creating a "kit" that the user can program, these 16 pads can be played at once to help recreate the sounds of a real drum-kit. This design project aims to do away with these physical pads by instead mapping user body actions to musical samples.

Figure 2 – An MPC2000 drum machine.

They also have the ability to output digital controls to computers or other audio equipment for further storage and manipulation. The benefits of abstract samplers are that they allow users to quickly and easily create drum loops. They also allow the user to work with music samples in an intuitively. However, the drawbacks of abstract samplers are the high cost and size factor, making them a hassle to move around. Based on user research and interactions with the target market, users of abstract samplers found the 4 by 4 square pad layout very useful. Project MUSE will use the pad layout, as shown in the User Interface mockups, to store the samples and sound patch in each square pad.

Mobile apps

Several apps are available which enable quick music production from the home studio. BoomClap [3] is an app available for the iOS operating system, which lets the user control a drum machine by creating sounds that the app recognizes as one of four discrete tones. The app uses an algorithm to separate the sound into four categories, saving its interpretation of the users sounds as a drum track.

Vocal Beater [4] is a similar app that attempts to map human vocal “beat-boxing” to a drum track, dividing it into the individual drum sounds (kick drum, snare drum, hi-hat, tom-toms). The final drum patterns can be emailed to the user for further editing with studio equipment.

Figure 3 – The Vocal Beater user interface.

Each of the apps discussed above lack flexibility in recreating the user input with a variety of sounds or inputs. Furthermore, all the apps do not seem like a professional tool, more of a gimmick, and lack compatibility with studio software. They are also not reliable and users have experienced significant issues with them. However, due to the nature of apps, they are an affordable solution and assist producers on the go. Similarly to these music apps, Project MUSE is constructing a software solution for mobility and low cost, as well as employing similar pattern recognition techniques, with significant improvements and added features.

Studio software

Studio software such as Apple's GarageBand [5] and Logic Pro [6], Cakewalk SONAR [7], FL Studio [8], Ableton Live [11], among many others provides the tools of a studio in a software environment. Artists can record audio, edit and add effects, combine audio samples, or use on-screen tools to create music notes for virtual instruments using the interfaces provided. Some studio software such as Ableton Live 9 allows the user to input audio with a microphone and it will programmatically create a drum track by utilizing pattern recognition techniques [11]. However, the results are not completely reliable and its usage is limited to the Ableton Live 9 Suite, which costs $450.

Figure 4 - Ableton Live 9's user interface.

Project MUSE recognizes that studio software is the essential tool of today's musicians, which cannot be replaced, and is a required component for layering multiple tracks (one for each instrument) to create a full song. Therefore, the solution that must be created should integrate with studio software - Project MUSE's final prototype will integrate with studio software, through the VST plugin protocol. In addition, the MIDI output track can be directly imported into any studio software; multiple MIDI tracks can be easily edited, within any studio software, to create a full song.

Drawbacks of existing products

While many products exist for the current market, they are limited by one of several factors including steep learning curve, cost, the need for extra equipment and musical competency.

A high learning curve is often required for even the simpler pieces of studio software so that the user can arrange the various parts of the song into a complex work of art. There is the additional cost of learning how to play a synthesizer or every specific musical instrument (drums, guitar, etc.) required to perform each part of a song. Similarly, there is a vast array of knowledge that must be acquired in order to use samplers and drum machines, even to create simple songs with a few instruments each. Cost is another drawback, the software products can cost up to $500 [9] or more for the more complex and powerful virtual studio tools. Individual hardware samplers or

sequencers, which may reduce the learning curve or complexity of the song, can cost several hundred dollars each [10] and may not represent the entirety of what needs to be purchased. Additionally, many components are often required to produce one piece of music - if an amateur performer wishes to record a reasonably-complex song for eventual production, a good-quality microphone, audio recording interface (to connect the microphone to the computer's audio inputs), and studio software all need to be purchased individually - not to mention the individual instruments which may need to be played manually. Extra equipment is also usually required, reducing portability and the quick creation of music when inspiration strikes.

Based on discussions with the target market, musicians often find it difficult to "sketch" out their idea whenever one comes to mind. To record musical instruments playing, quiet environments are essential, which often requires the creation of specialized microphone booths in order to record. Individual instruments, amplifiers, the associated cables and interfaces, as well as hardware interfaces for drum machines and synthesizers all increase the physical weight and size of the equipment which is required to create even a simple song. While software has severely reduced the amount of equipment that must be used, it can require the mastering of a multitude of interfaces and techniques in order to be used effectively, and does not possess the same sensory bandwidth as hardware like touch and spatial sense.

Finally, the production of music requires the mastery of multiple skills, as music generally has multiple instruments and steps involved in its production. A multitude of sound engineers, performers and producers may be involved in the production of a single song, and it can be difficult for a musician with an idea to produce a song without the help of others or knowledge of every facet of music production.

The solution space has a vacancy for a music production tool that requires little knowledge of extraneous skills and few extra components. The solution should incorporate the benefits and improve upon the negatives of each tool in the current state of art.

Academic research

In addition to research and development involved in the formation of aforementioned commercial products, some academic research has been done in the immediately relevant area of translating vocal performances into music. One particularly extensive example of this research is a PhD thesis by Dan Stowell [13], wherein a plethora of approaches to translating percussive and melodic vocal performances into musical data are tested and various feature sets and classification methods are explored.

Furthermore, extensive research has been done in tangentially relevant areas such as speech recognition, the results of which have proven useful for general audio feature extraction and classification.

1.3 Project goals and objectives

The ultimate goal of the project is to allow musicians the easiest tool possible to go from idea to execution at the beginning of the music production process. The principle component of this goal involves the interpretation of one-dimensional (1-D) sensor signals to be classified and translated to musical output sounds based on associations set by the user beforehand. The user's input will be generalized to allow for a variety of 1-D input types, including audio input, accelerometer and gyroscopic position change data, or vibration. It will be designed such that the core functionality will be compatible with a wide variety of inputs and outputs as the landscape of interactive technology grows.

1.4 Design requirements

Based on the problem statement, a set of requirements was formed. This list includes needs important to both the end user and the developers who will create the system.

Performance

The system must be fast enough to operate in seemingly real-time, and must be run by currently owned laptops and/or smartphones. This is to keep up with the pace of the music being created, and, generally, software should run fast enough such that it seems the user does not have to wait for it to perform its tasks.

Accuracy

The system must interpret user input and consistently translate it to the correct user-intended output. This is the most important parameter, as good input should not lead to a flawed output. If the output does not reflect the user's desires, then the system is not achieving its purpose.

Ease of Use

The system must be easy to use such that a novice can start creating music immediately with virtually no learning curve. This eliminates the aforementioned difficulty associated with learning a new instrument, device or software tool.

Affordability

The system should be inexpensive such that cost is not an obstacle to the musician. This makes our solution more practical than expensive music production devices.

Maintainability

It must be easy to make changes to the software as deemed necessary, such that it can be developed without additional difficulty over the course of our project and perhaps in the future as more inputs and outputs are added.

1.5 Partnerships

Project supervisor

Dr. Hamid Tizhoosh, Systems Design Engineering Professor, has offered guidance in the pattern recognition and machine-learning component of this project. He is also a professor of Algorithms and Data Structure courses. Furthermore, his knowledge of intelligent systems has contributed to the success of this project.

MBET

During the 4A term, the team members of Project MUSE pitched their idea to a variety of MBET students. One of them, Rachel Bartholomew, is a professional music producer and has released commercial music. She was consulted, after the pitch, and

provided feedback from the perspective of a potential user. Her feedback helped shape the product and she contributed ideas regarding the needs of users and the gaps in the market. She also connected the team to other music producers who have shown interest in the initial prototype.

FELT Lab

The FELT lab, located in REAP (Research Entrepreneurs Accelerating Prosperity) building in St. Jacobs, Ontario, focuses on fostering next generation multimedia technology. FELT lab has provided an opportunity for the team to learn from industry researchers and potential users of the system. By attending a Lunch and Learn for the opening of the sound lab, the team was able to gain insight into the current state of the art, and was able to validate the project idea with researchers. FELT lab has also shown interest in Project MUSE and the project was demonstrated at a February Lunch and Learn, which functioned as the outreach course requirement. Further work on the project may involve assistance from FELT Labs in the areas of advisory and funding acquisition.

Customer panel

A panel of potential customers was formed based on the target customer market. The panel consisted of professional music producers, amateur experimental musicians, technologists, and individuals with a slight interest in music and no training. Individuals who were part of this customer panel include Gaurav Dayal (a professional music producer, Cam Mckittrick (a Music and Technology Professor at the University of Waterloo), Dr. Hamid R. Tizhoosh (a Systems Design Engineering Professor and the Project Supervisor), and Rachel Bartholomew (an MBET Student and professional music producer).

2.0 Engineering Design

Project MUSE is divided into several sections that encompass the major portions of the project's design. First, the training samples are split into different notes, then each note has its characteristics measured and a vector created which represents the note in the feature space. Finally, the vectors are passed into a classifier to train it to recognize each separate class. The whole process is then repeated for the input audio to be tested against. A full solution diagram is illustrated below.

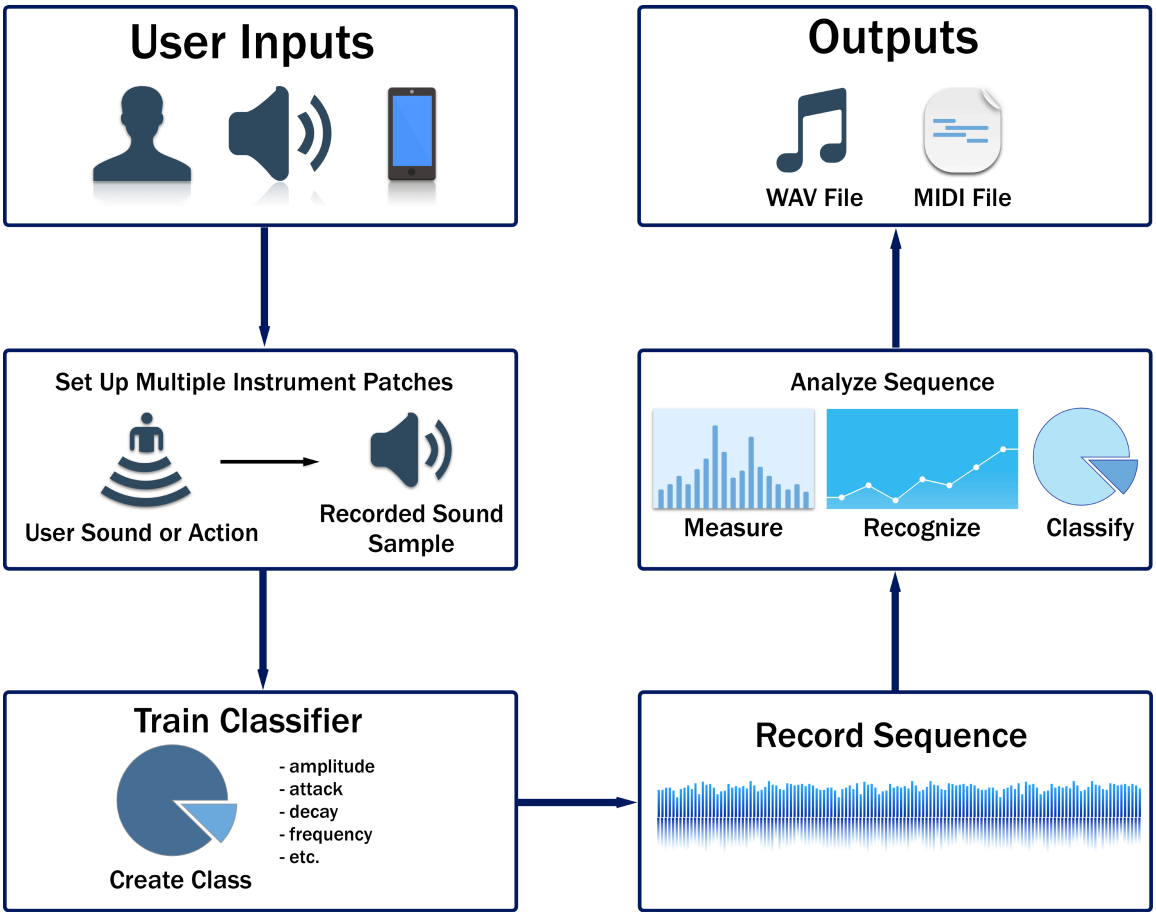

Figure 5 - Solution Diagram

2.1 Note detection

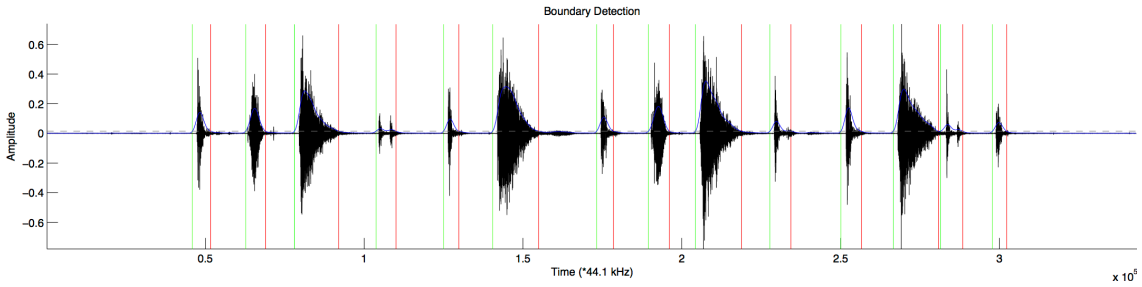

Figure 6 - Successful boundary detection on an input audio signal.

Determining where the notes are in the original signal depends on a series of factors, and was a large portion of the work involved. This process involves several steps, including determining where boundaries are, splitting notes where the boundaries overlap or are in too close proximity, or fusing many boundary detection results into the "real" note boundaries.

Boundary detection

Many of the classification errors during the initial stages of the prototyping process involved problems with detecting the boundaries of notes such as treating multiple notes as one, adding extra notes where there was significant spikes in background noise, and ignoring notes entirely. In the end, a three-pronged approach was taken to firmly establish a note's presence.

Amplitude thresholding

A full-wave rectification was performed on the underlying audio signal, and the resulting data was smoothed using a Gaussian smoothing filter to better characterize the note's shape, while avoiding artifacts such as ringing or blockiness. This gave an approximation of the amplitude across the length of the signal, which could be compared to a noise floor. Any contiguous section of the signal with an amplitude value larger than that threshold was considered a note.

Frequency distance

Using a windowed RASTA-PLP transform (described in more detail in Section Blah, Frequency Characteristics, below), the signal's spectrum was calculated and a 25-

coefficient frequency vector was produced across the signal's length. Using the recorded background noise, the same vector was produced to represent the unwanted or noisy components of the frequency. Finally, the Euclidean distance was measured between the background noise and the signal; contiguous portions of significant distance (above a threshold level) were considered notes.

N-Grams

The same vectors from the RASTA-PLP transform were taken from all the training data and compressed into audio "strings" by merging similar frequency vectors together. This string represents each note in a temporally agnostic way, only taking into account the order of distinct parts of the note spectrum. For example, a kick drum sound has an initial high-frequency-dominated "attack" section comprised of sibilance or other vocal characteristics of the note (such as a "b" or a "d" sound, common for a kick drum). Immediately following that is a longer lower-frequency-dominated section, which decays in amplitude until the end of the note.

Using a common Markov Model n-gram search technique and testing similarity based on a thresholded Euclidean distance, common strings of these notes were found in the underlying signal. The boundaries of these signals were treated as boundaries of notes.

Boundary fusion

These three techniques gave three slightly different representations of the boundaries of the signal, and had to be merged in some way.

Local merging

Conveniently, each boundary technique discussed above gave a Boolean vector in MATLAB with true values for indices that it considered notes in the signal vector. Because the three features which were used for boundary detection (significant amplitude difference from the noise floor, significant vector difference from the noise spectrum, and part of a chain of frequencies which has been recognized) were all considered indicators of notes, a simple logical AND between the three note-detection vectors would give a vector of portions of the original signal which pass all three tests.

Further techniques

All three techniques are based on some kind of thresholding to convert a vector distance or difference into a simple Boolean value, which simplifies the actual distances. To better preserve the subtle variances in distance, a probability could be determined (instead of a Boolean value) which could be fused based on element-wise multiplication. Elements with high probabilities (above a final threshold value) of having all three characteristics could then be considered notes.

Note splitting

Once boundaries have been established, it is common for notes to be joined together; this happens for a variety of reasons, including close note proximity (temporally and spectrally) and errors in vocal performance. There may also be small variations in contiguous notes where splitting is desired - especially with tonal interpretation, where a note may be held but change pitch. This means that it is useful to be able to perform a secondary analysis on the note boundaries and determine if there are points at which the note could be split.

Significant valleys

Notes in close proximity, which are nonetheless distinct, can be split using amplitude-based methods. The smoothing on note start and end times is fairly aggressive, which has a tendency to smooth over large valleys in the rectified notes. Using a less aggressive smoothing, the signals are run through a valley-detection algorithm that looks for portions of the signal that are under a minimum threshold from the lowest peak on either side of the valley. If such a significant valley is discovered, the note is split.

Pitch changes

For tonal notes, the underlying MIDI signal is based on discrete semitones, where each note is “rounded” to the closest semitone. Using the existing pitch interpretation function, the note is windowed and its pitch across the length of the note is determined. If the pitch changes between two windows, the note is split at the point between them.

Interestingly, this technique gave insight into the complexities of the human pitch-interpretation system, which has many differences from pure mathematical pitch. The

human brain has a tendency to ignore small variations in pitch once the note has been established, whereas the pitch-detection function can often be brutally honest about the effects of vibrato singing (adding texture to the note through amplitude oscillations, which can change the frequency), the effects of speech phonemes (as opposed to pure sinusoidal tones), and simple flat or sharp singing. Some work has been done to combat these variations (such as increasing the window size for the first portions of the note, where the phoneme-variation is highest), but there are many other possible additions to the technique; they are discussed in section 4.2 Improving melodic performance interpretation.

2.2 Classification

To perform classification, measurements were taken on the individual notes, based on features that had to be chosen to try to best characterize them while keeping them distinct from other notes. A number of possible features were considered, and their performance was evaluated based on accuracy, with some being chosen over others.

The classifier used was a simple Naive Bayesian implementation, chosen for its simplicity. While there could be benefits for performance and efficiency in choosing another classifier implementation, which was left to a related project in the SYDE 522: Machine Intelligence class.

Note characteristics

Basic characteristics of the note, including its root-mean-squared amplitude and its overall length were included, with the hopes that there would be some kind of trend that differentiated the various notes from each other. For example, crash cymbals tend to be rather long and loud, whereas hi-hat cymbals are usually shorter, despite having similar frequency characteristics.

Amplitude characteristics

Musical notes are often divided into different sections relating to different parts of the sound's characteristics. There are a variety of common patterns, including the ASDR (Attack, sustain, decay, release) envelope in use by many MIDI instruments.

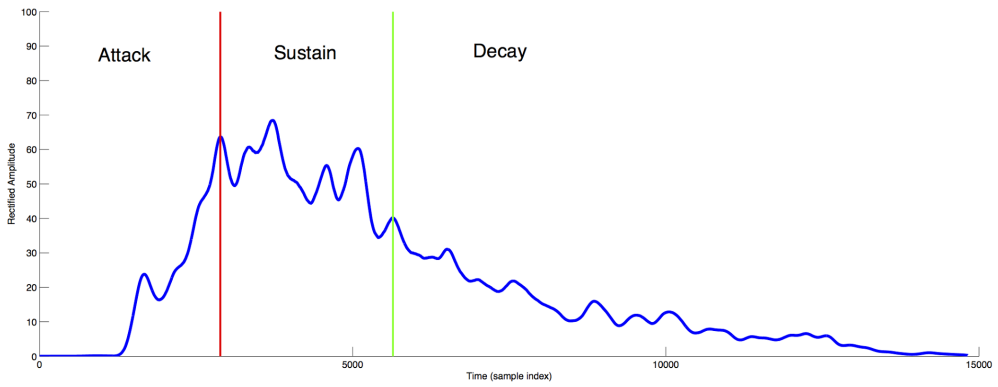

Figure 7 - The amplitude characteristics of a crash cymbal. The blue line is the smoothed, rectified wave.

For Project MUSE, a simpler envelope incorporating only attack (the section before the first major peak), sustain (the middle section of the note) and decay (the section after the last major peak) to ease calculation. To ensure the values were not dependent on the length of the note itself, the ratios between the various sections were used instead of their absolute values.

Frequency characteristics

Early on it was obvious that some kind of frequency component was going to be needed for the analysis, as it is one of the two main fields in signal processing and is a large component of audio analysis. Several initial attempts were made based on raw Fourier coefficients to try to characterize the signal. The highest four peaks of the transformed signal were used as an attempt to capture the fundamental frequency and some of its harmonics. However, that failed to properly capture the characteristic of the sound, as the amplitude of the lower-frequency signals were much higher, so it was more biased towards that low end.

While researching alternative approaches, a combination of two transforms called the Relative Spectral Transform - Perceptual Linear Prediction (RASTA-PLP) transform was discovered, which was designed for speech recognition through audio [12]. Perceptual Linear Prediction is a technique that changes a Fourier spectrum to eliminate variances between speakers and preserve the common speech information. The Relative Spectral Transform is another technique that eliminates DC offsets and high-frequency noise

(such as static or other interference) by bandpass-filtering the signal. As the input signal to Project MUSE was recorded speech audio, this technique meshed perfectly with its goals. In the end, the coefficients of the RASTA-PLP spectra and ceptstra (the inverse Fourier transform of the logarithm of the spectrum) were added to the final measurement vector.

Principal Component Analysis

Principal component analysis was used to try to improve classification rates by increasing the between-class variance, and reduce computation times by performing dimensionality reduction using the results of the analysis. The training data was run through principal component analysis, and the principal components were determined, along with a transform for incoming testing note vectors to be transformed to the same space. Additionally, the variance of each dimension was returned and any components that contributed a variance over a threshold value were used in the final measurements.

This led to the discovery of one of the flaws of principal component analysis, specifically its sensitivity to outliers. Unfortunately, due to the presence of such outliers in the training data, the variance was highly overstated and the training data space was actually severely compressed, leading to a dramatic decrease in accuracy. In the end, PCA was temporarily in favour of using the raw data, which gave an overall increase in performance, until a more robust handling of outlier data can be undertaken.

2.3 User interface

A user-interface was designed for the iOS application of Project MUSE. An in-depth design study for developing a compelling user-interface was completed as part of SYDE 542 Interface Design in-class assignments. The user-interface was developed only for the percussive track type for the initial version of the mobile application.

First, a requirements phase was completed including user persona analysis and work domain analysis. Secondly, a wireframing phase was completed including the development of a navigation plan. Third, a design review was completed which involved reviews of the wireframes by both novice users and stakeholders. Fourth, a wireframe text was completed in a lab style format involving objectives, method for testing, and

results section. Finally, based on the results from the wireframe test, a detailed mockup was designed. A summary of the user-interface study is provided below.

Requirements phase

Requirements were gathered using two distinct methods: user persona analysis and work domain analysis.

User persona analysis

User persona analysis involves describing each facet of a potential user of the tool, including demographic information and a detailed description of the user's situation. An excerpt of one user persona is illustrated below.

Name: James Blake
Age: 20
Location: Waterloo, Ontario
Marital Status: Single, no children
Occupation: Student, Electrical Engineering
Income: Current not employed, $500/month from parents.

James is as tech-savvy as any regular 20 year old, with a Facebook account, the latest iPhone, and a mac. He is studying electrical engineering at the University of Waterloo. James enjoys playing his guitar for his friends, and wants to record a song for his girlfriend. He doesn't know how to play any other instruments. He doesn't have much time other than a couple hours a week since he is on campus most days from 9am-10pm due to his demanding school schedule. He only knows how to use the basics of Garage Band, mainly recording with a microphone and editing length. He lives in a small one-person dorm, with not much room for other equipment or instruments. He wants a drum-backing track for his guitar-based song with some small piano pieces. He knows the drum pattern will be simple, but he is having difficulty programming a drum pattern in Garage Band.

Work domain analysis

A work domain analysis investigates the system boundary, purpose, principles, processes, components, and attributes. The method is focused heavily on abstracting the user from the application and analyzing the user interface from the perspective of the underlying system. The following requirements were gathered from work domain analysis.

1. Intuitive to use – user does not need to consult help
    a. Setup multiple sounds easily
    b. Record sounds easily
    c. Record a noise profile easily
2. User should not be overwhelmed by menu screen
3. User should not have any mis-clicks due to buttons sizes
4. Microphone should be need to be configured once
5. Record sounds (95% record rate)
6. Analyze training sounds with sound feature analysis (90% correct)
7. Analyze sounds pattern accurately with pattern recognition (90% correct)
8. Playback of sounds (100% successful playback of sounds)
9. Conversion to midi or wav file format (100% successful conversion)
10. System should not crash (<5% crash rate)
11. Quick pattern and sound analysis (< 20 seconds)

Wireframing phase

Based on the requirements gathered in the previous phase, a critical scope of the interface was determined. The critical scope allowed the navigation plan to be developed.

Navigation plan

Based on the requirements gathered from the requirements phase and after outlining the critical scope of the project, a simple navigation plan was developed. The plan outlined movement between 4 main screens: Home, Main Project screen, Help, and the Sample Edit screen. Each of the screens are illustrated below.

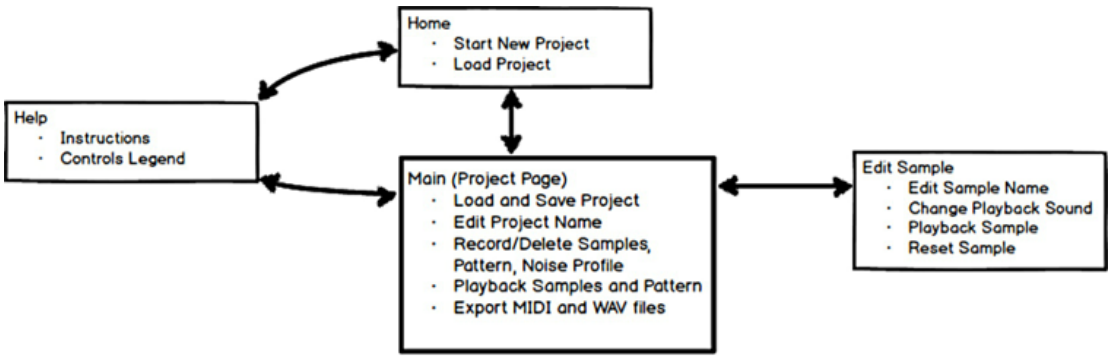

Figure 8 – Navigation Plan

Wireframes

Using the information gathered up until this point, wireframe mockups were designed in Balsamiq Mockups, illustrated below are a few of those.

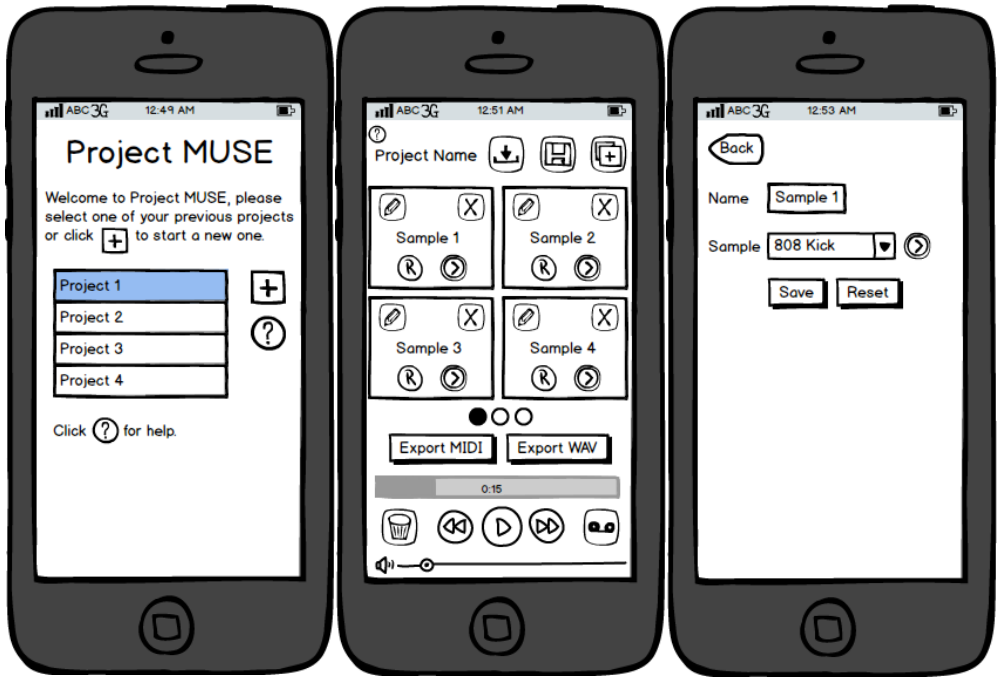

Figure 9 –Wireframe Mockups

Design review

An informal design review was completed with both novice users and stakeholders, for which the wireframes were linked together in order to emulate a functioning application. An excerpt of comments from both the novice user review and stakeholder review are outlined below.

Novice user review

The novice user that reviewed the wireframes had no previous musical training or familiarity with music based software. Also, the novice user had no specific objective or reason for utilizing this application.

- Not very intuitive, difficult to use without instructional page of tutorial
- Hard to determine what buttons correspond to which functions
- After going through the help page, the user was able to understand the flow of actions, but the end goal was still not clearly described
- The user enjoyed the simplicity of the design after all elements were explained
- The user liked the flexibility in recording any sound in the application
- The user also liked the confirmation alerts before any changes were made
- The user suggested that more effort could have been made to explain each function rather than having the user rely on trial and error
- The design could be less cluttered if the buttons were spaced better or if they were shaped differently

Stakeholder review

The stakeholder user that reviewed the wireframes was musically training, being a guitarist with a specific use for this application which is to create a drum track for a song.

- Very familiar to traditional music production software
- The user immediately knew how to start a new project, record sounds, and patterns
- The recording of the noise profile was not obvious; there was no explicit alert to do this until the user hit “Export”
- The user appreciated the minimal screens, flexibility in selecting any sample, and confirmations before all major actions
- More flexibility was desired for editing the pattern and increasing or decreasing the length of each recording

Based on the feedback from the design review, some changes were made to the wireframes, and some were kept for development of the full mockups.

Wireframe test

A full wireframe test was conducted in a lab style format. The objective of the wireframe test was to test the intuitiveness of the user interface, identify key areas that can be improved and receive feedback on which parts of the interface were enjoyable and done well and which were not. The method and results of the wireframe test are summarized below.

Method

The method for testing the objective was to find volunteers to try out the product (in the form of wireframes) using a prescribed script found in Appendix A in order to standardize the experience between all users. The users were judged on how well they could accomplish each task based on a scale ranging from easy to difficult. In addition, questions regarding the user experience of the product were occasionally asked to understand underlying problems or achievements. The data was collected in a way to determine common mistakes or misconceptions with the wireframes.

Results

From 5 users that were tested, it was determined that tasks involving single commands were easily completed, however tasks such as recording (involving more than one command) were not easily completed. Furthermore, it was discovered that when the users were first instructed on the tasks, subsequent tasks were performed effortlessly. Therefore, it was found that instructional tips or an initial tutorial for the interface would offer users the most benefit.

Full mockups

Based on the information retrieved from the informal design review as well as the wireframe test, full mockups were developed. However, only a few suggestions were implemented due to time constraints. The full mockups were designed for iOS 7 iPhone devices taking into account font, color, and gestures as close to the final working application as possible. A few of those full mockups are illustrated below.

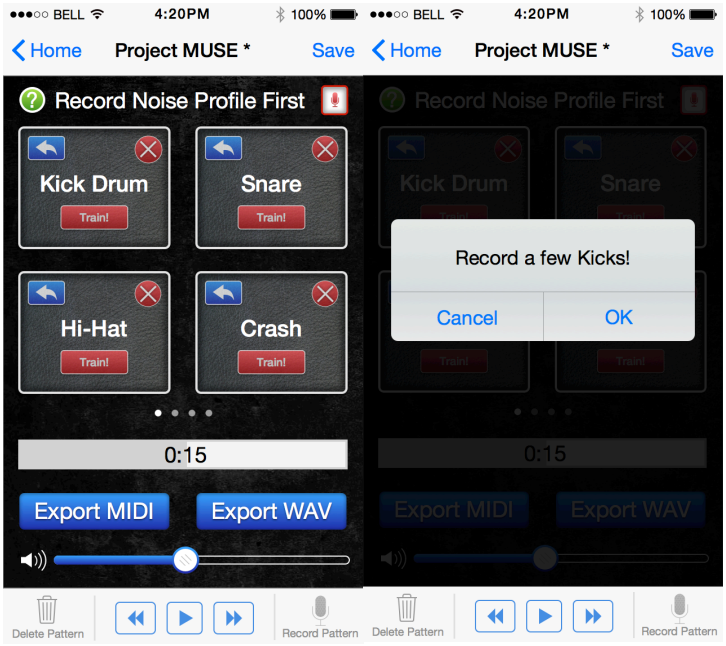

Figure 10 – Full Mockups for iOS 7

MATLAB graphical user interface

Aside from designing the mobile user interface, a full working graphical user interface (GUI) was developed in MATLAB that complimented the prototype. This interface, however, was not developed with significant study or process, but as a quick way to test the code in MATLAB. Some ideas and concepts were taken from the research conducted during the design of the mobile application, but in rudimentary fashion. The objective for the development of this interface was to capture all developed functions in one screen. The MATLAB GUI is illustrated below in *Figure **11***.

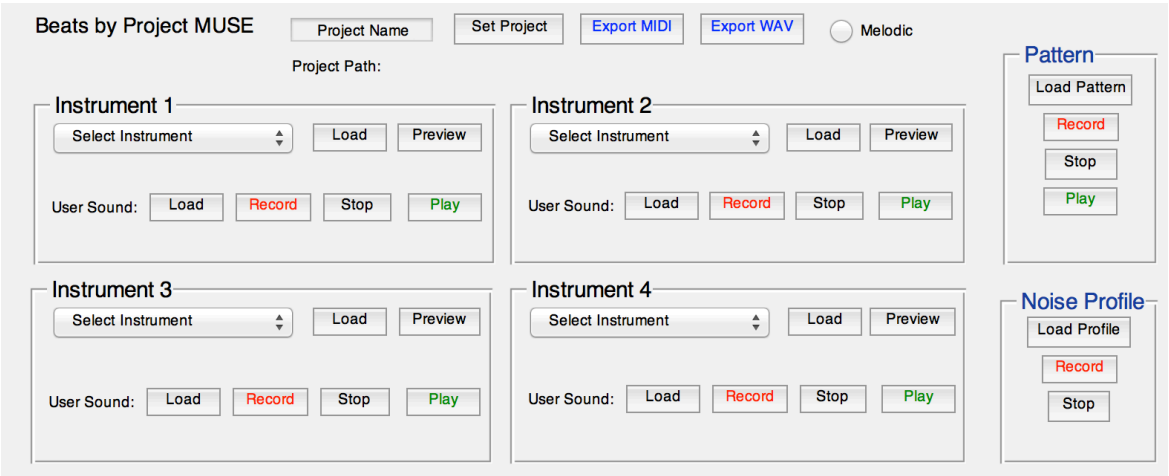

Figure 11 - The MATLAB GUI demonstrating the prototype interface.

Notable challenges and resolutions

There were many challenges holding the team back from producing an excellent user interface. First, the system itself is very complicated and had to be thoroughly analyzed to ensure that all areas were being accounted. In order to accomplish that task, the critical scope had to be recognized as done in the work domain analysis phase, with significant requirements gathering as a prerequisite. Furthermore, the decision to only focus on the percussive track for the mobile user interface allowed for further simplification of the design variables.

Next, there was significant challenge in developing the user interface in a way that would be logical to both novice users and stakeholders. During the informal design review, each user group had very different sentiments towards the layout of the interface. The novice users wanted a simplified design with more tool-tips and tutorials, while the stakeholders preferred a more complex interface with finer control and flexibility. In order to appease both target user groups, the user interface was simplified, yet the controls remained available in other screens for more advanced users. In addition, the interface was designed to prompt a first time user with a tutorial that outlines each feature of the application alongside common use cases. After the tutorial is completed, it will remain hidden for subsequent uses of the application, with access to the tutorial in the help section for review at any time.

Finally, a major challenge was conducting wireframe testing in an effective manner. The wireframe script had to be developed in a manner that clearly outlined the expectations and methods to capture feedback effectively. The script was initially designed without expectations and open feedback to capture all possible responses, however it was further modified with a specific feedback structure and set expectations. The final result was a mobile user interface that met the requirements of both novice and advanced users while still retaining an intuitive experience and flexible functionality.

3.0 Design Testing and Validation

With a classification problem such as this, a measurable method of evaluation is extremely important to determine the algorithm's performance. The overall performance was measured to be 89.20% for atonal (percussive) recordings, and 39.78% for tonal (pitched) recordings; the discrepancy reflects problems with the accuracy metric, which is discussed below, in Section 4.6.1, Expanding the test suite.

3.1 Accuracy metric

The metric used for the accuracy had to be developed to reflect the perceived accuracy, rather than the mathematical accuracy of the underlying data. It rated differences in note boundary (start and end) times, giving each one a normalized accuracy rating as compared to the closest corresponding boundary time in the transcribed file. Extra or missing note boundaries (from extra or missing notes due to improper note detection) were given accuracy ratings of zero. The rest of the notes are rated according to their difference from the closest "true" note in the manually transcribed file, which is passed through a normalized Gaussian probability distribution function, where small differences were ignored with a rating near one, but large differences approached zero.

The accuracy metric also incorporated sound classification correctness, given as the percentage of correct classifications. To combine the different accuracy metrics (start, end, and classification), a weighted average between the three was taken. The rectified profiles of the notes themselves resemble exponential or gamma functions (as shown in the attack/sustain/decay model) so the perceived accuracy is more dependent on the start time than the end time, and the weights are adjusted as such.

3.2 Test suite

To test the accuracy of the algorithm, a set of beat-boxed patterns were recorded and manually transcribed into MIDI format. To ensure the tests represented a wide variety of use cases, the recordings span different microphones – including mobile phone recordings, laptop recordings, and higher-quality condenser microphones – and in recording environments from stairwells with lots of echo effect, rooms with humming

equipment, and crowded hallways. A variety of users were also recorded to ensure performance across different vocal characteristics and accents. Additionally, the beat-boxes were performed at a variety of speeds; all of this variation was to either ensure performance across the variables, or track performance degradation if it changes.

Once the files were assembled and manually transcribed, an automated testing program was created to be able to consistently verify the accuracy of the program across the beat-boxing patterns. The program searched for transcribed audio beat-boxing files and compared the MIDI output from Project MUSE to the manually transcribed file, giving a rating from 0 to 1. Finally, the test suite calculated the mean of the accuracies from the individual files as the overall accuracy.

3.3 Accuracy characteristics

It was desirable to know the full characteristics of the accuracy of the overall algorithm so that it could be improved by controlling certain variables that were linked to increased error. From intuitive knowledge of how the performance was affected, a few areas were explored.

Beatboxing speed

Owing to the problem of notes in close proximity, it was thought that the effect of closer average note proximity might be inversely correlated to accuracy. To test this, the sample's speed was calculated as the median of the distances between the start times of each note. Each sample's accuracy was plotted versus its speed to determine if there was an overall trend. The results are shown in *Figure **12***, below.

Overall, there seems to be a general trend downwards as the number of notes per second increases, but there was not enough data to draw a full conclusion. More samples should be recorded and transcribed to evaluate this conclusion.

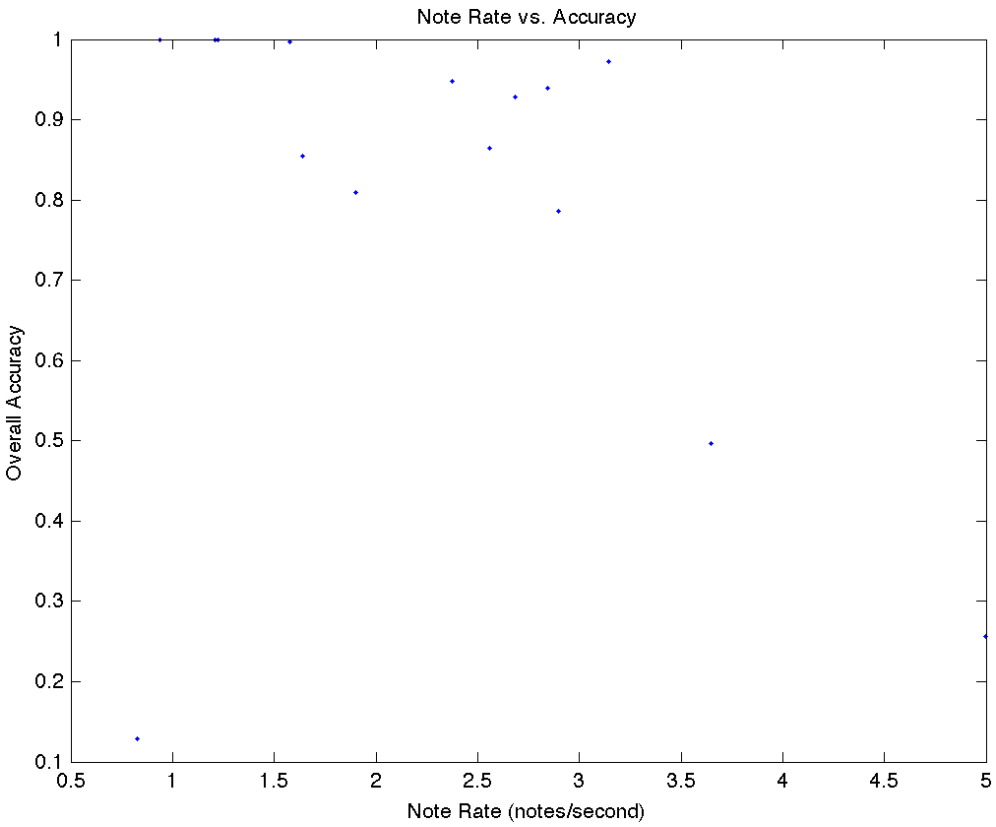

Figure 12 - The note rate vs. overall accuracy.

Overtraining

A graph of training data samples per class versus accuracy was created to determine whether the model was being overtrained, as well as to attempt to determine a point after which adding new testing data would not improve accuracy. This could help guide the initial training stage in the user interface, by finding a recommended number of training samples to record.

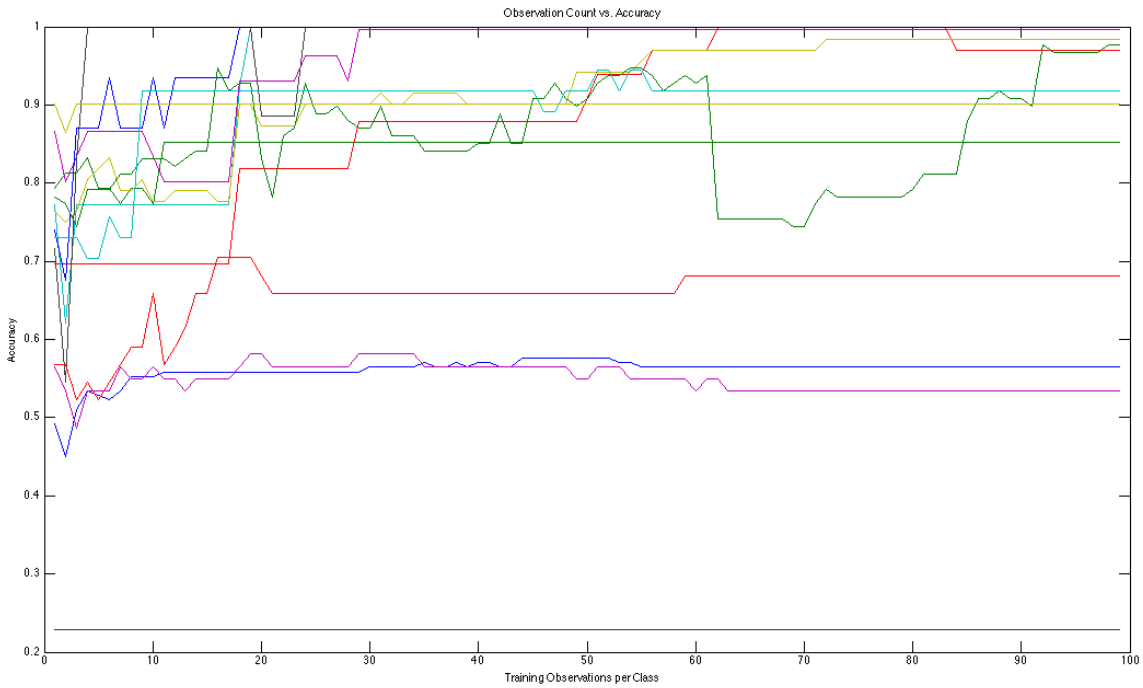

Figure 13 - The overall accuracy, compared to the number of training observations per class.

From the data shown in *Figure **13***, it can be seen that, as expected, the number of training samples is roughly correlated to overall accuracy. After around thirty training samples, most samples have stabilized and are near their eventual accuracy ratings; after sixty training samples, almost no improvement can be found. While this test could also benefit from more testing samples, the overall trend can be intuitively confirmed and most samples will be correctly classified at between 30 and 60 training observations.

4.0 Recommended Design Modifications

4.1 Classifier research

The Naive Bayes classifier was chosen due to its simplicity, but further research is being done into what classifier performs more optimally and increases the overall accuracy. As part of a SYDE 522: Machine Intelligence project, different classifiers are being evaluated for their effectiveness, and characterized across different levels of training and beat-boxing speed. Once this research is complete, the Naive Bayes classifier can be replaced with whatever new classifier has the best performance.

Additionally, more research needs to be done into what features contribute the most between-class variance and pruning those that do not. Similarly, research can be done to add more features to give higher performance overall.

4.2 Improving melodic performance interpretation

Melodic performance interpretation was added later in the development process, and as such has not been fleshed out enough to form a viable product. Testing metrics show a low accuracy rating, and it is immediately observable that the natural fluctuations of the average human singing voice result in an excess of output MIDI notes fluctuating around the desired pitch. The implementation of a subtle “auto-tune” to smooth these rapid fluctuations into single notes is desirable. This would involve dynamic adjustment of the size of the window used by the boundary-detection function for melodic performances.

4.3 Handling overlapping notes

One of the largest challenges of the project was the handling of overlapping notes. In melodic performances these would be chords, and in percussive performances these would simply be overlapping drum sounds. Stowell made significant progress in the handling of these problems [13] and the addition of these capabilities to the system is highly recommended. For percussive performances, the “feature stacking” method can be employed, which involves adding new classes for each possible combination of input classes to be considered by the classifier by simply creating additional feature sets from of the sum of the features of the combined classes. For melodic performances, chord

interpretation is a problem that has been researched extensively as well, and is possible via thorough Fourier analysis.

4.4 More applications and environments

Programming languages and environments

Recommendations for furthering this project as a commercial application involve the porting of the system to one or more programming languages more conducive to commercial products such as a VST plugin as well as Android, iOS and desktop applications. The most appealing languages for these applications would be Java, C/C++ and Objective-C. Java in particular would be best for an Android app, though it could be used to create a multi-platform desktop app or even a VST plugin through the use of an open-source wrapper. C or C++ would be the best option for writing a VST plugin, and in addition modules within the code could also be used within Android, iOS and desktop apps, speeding up the multi-platform development process if this route is chosen. Finally, Objective-C would be necessary for writing iOS and Mac OS X desktop apps - either to write the full app, or to interface previously written C/C++ code with Apple devices.

Even within the arena of rapid prototyping, it is advisable to switch to a new language. Python and its vast world of modules allow even more signal processing and machine intelligence capabilities than MATLAB. In addition, Python is a simple, well-established object-oriented programming language which would be useful for both rapid prototyping and commercial applications in areas such as desktop apps and interfacing with new input devices (such as Leap Motion [14], Microsoft Kinect [15], and possibly Thalmic Labs Myo [16]).

Inputs and outputs

As the core functionality approaches optimal levels, additional input and output formats should be explored. Since the core translation capabilities are built as a module, which can interface with any input and output module, integrating new features in this manner will be a simple process of building new modules for the new input or output formats.

Possible additional inputs include table drumming, body drumming, smartphone accelerometers, generic controllers such as Leap Motion, Thalmic Labs' Myo and Microsoft Kinect, or even biosensors such as NeuroSky's ECG and EEG solutions. The user experience for each of these input devices would be the same, as users would go through the same classifier training process wherein they execute multiple instances of each input and map them to their desired outputs. The versatility of the MIDI standard is such that inputs can be parsed and interpreted as desired to create a MIDI output.

Possible additional outputs include applications MIDI is already commonly used for such as lighting control, clip cuing for DJs, dynamic animation control, and even video games [16]. The versatility of the MIDI standard allows it to be a controller for any sort of electronic system in whatever manner is desired by the user.

4.5 Real-time operation

The system would also benefit greatly from the capability to operate in real-time rather than interpreting complete performances after their conclusion. The largest challenge in this area is operating speed, the ability of the system to react fast enough such that there is no noticeable delay. Porting to a lower-level language (such as C/C++ or Objective-C) will likely eliminate much of the processing overhead, though the computational efficiency must be made a high priority, an area that will require significant research on part of the developers.

4.6 Further exploring accuracy

A few ideas for further exploring accuracy were considered, but not implemented due to time constraints. A better characterization of the fitness function could allow for more intuitive understanding of the algorithm's performance, and more focused development to increase the accuracy of the project.

Expanding the test suite

It is recommended that the testing system is expanded upon to make the metrics more meaningful. The metrics provided for percussive performances are currently fairly useful for development, though further work should be done to increase the extent to which these metrics assist developers. In addition, melodic accuracy ratings must be

altered to better reflect the quality of the system's output in this separate space. Note accuracy is currently rated on a binary scale (0% or 100%), rendering missed notes as 0% accurate. Though this is useful for percussive performances where a note is simply classified correctly or incorrectly, results of melodic performances involve a varying degree of error. For example, a C being misclassified as C# (an error of one semitone) is much less severe than it being misclassified as an E (an error of four semitones). An error metric reflecting the nature of melodic classification in this regard is highly recommended.

Global training data

A multitude of training sets were used for the various testing examples described in section 3.2 Test suite but were isolated based on microphone and user to ensure maximum accuracy. While it was assumed that the differences between users and microphones may not be significant and there may be the possibility of combining training data into one "global" set in order to decrease the amount of training data each user would need to record. This would greatly benefit new users, who may even be able to use Project MUSE without any initial training data and obtain results quickly.

Speed tuning

There are a variety of parameters present in the algorithm that could be tuned to be more tolerant of higher-speed beatboxing, such as the smoothing window sizes or some of the tolerance values. More investigation is required to determine exactly which parameters are speed-dependent, and how they can be adjusted to maximize performance across varying speeds, but it is a potential avenue of improvement.

5.0 Timeline and Project Management

This project employed multiple project management tools and techniques, including Gantt charts for overall project planning and Github for storing feature lists and issues for each sprint cycle. Furthermore, agile methodologies and best practices were utilized for the duration of this project.

The specific style of agile that the team used was Scrum, in this type of development framework each requirement is analyzed and made into a "feature" with a specific number attached to it indicating the difficulty of the task (as agreed upon by the development team). A set of all features created a product backlog, from which sprint backlogs were compiled on a weekly basis. Sprint backlogs consisted of a list of features that should be built within a week's time according to their combined difficulty. The sprint backlog was collected as Github issues to ensure it can be accessed anytime, assigned to group members, and tagged to specific commits when completed. A screenshot of the issues list from Github pertaining to one week's sprint backlog is shown below as an example.

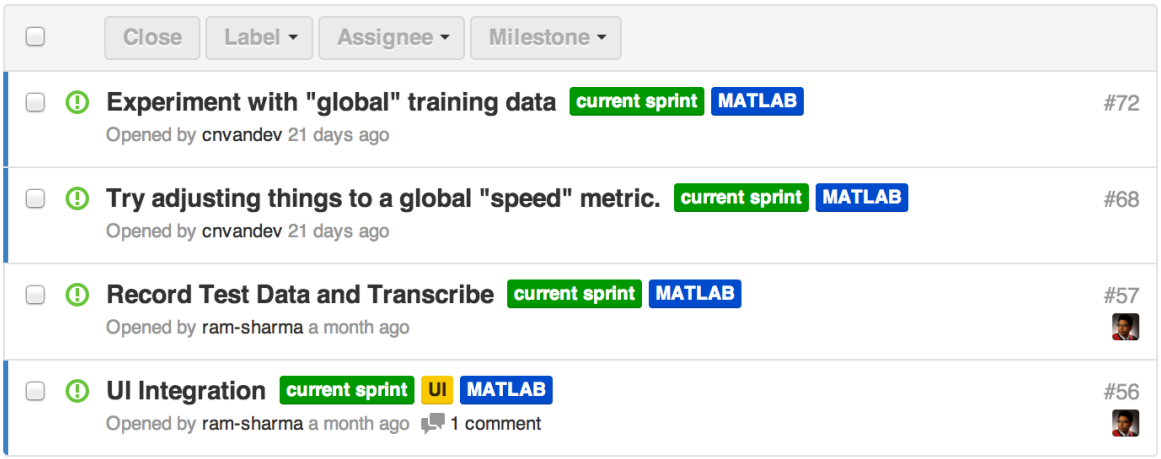

Figure 14 – Screenshot of Github Issues Page for One Week's Sprint

At the end of each sprint cycle, a working version of the software prototype was produced to ensure that the prototype is in stable condition at all times. The prototype underwent several iterations until it was deemed ready for final presentation at symposium. The Gantt chart below specifies the duration of each sprint cycle (or iteration) and major milestones for the entire project lifecycle.

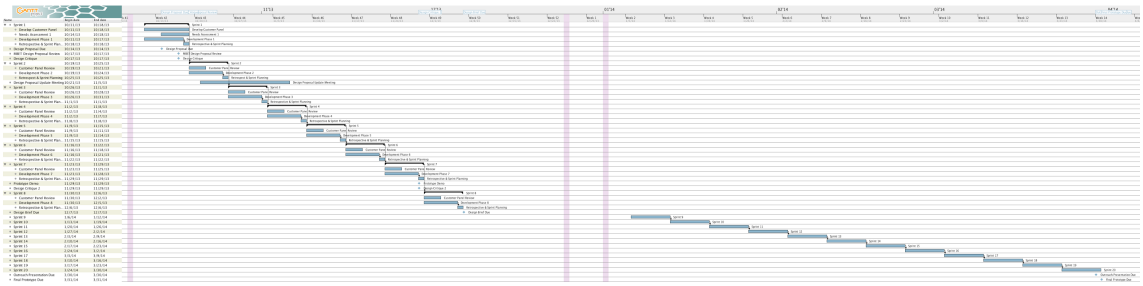

Figure 15 – Project Gantt Chart

The team also collaborated frequently in a group chat on Facebook, which allowed all team members to be notified of immediate project changes.

5.1 Critical review of project management

The team was able to complete 70 individually assigned features or tasks for the duration of the project, with an average of 5 per sprint cycle. The project was completed successfully, with the team meeting all set goals and deadlines. The team met every week, sometimes multiple times, to review the product backlog and set the sprint backlog for the week ahead. In addition, the team met with the project supervisor, Dr. Hamid R. Tizhoosh, on a bi-weekly basis to provide project updates, outline major challenges, and for general advice related to project progress.

However, there were are few hurdles throughout the project lifecycle, specifically related to features with major technical challenges, resulting in some changes to the project schedule and critical scope.

One major hurdle encountered early in the project was moving the codebase from Java to MATLAB. An initial decision was made by the team to develop the system in a Java environment; however, it was quickly found that there were many more advantages of using MATLAB to develop the initial prototype. Therefore, the team had to redevelop a few features in MATLAB resulting in a small schedule shift and reduced expectations of the final product. Overall, the situation was handled smoothly and the team worked together effectively to ensure that major milestones were still met.

Another major hurdle encountered by the team was related to estimating the time to develop certain features such as boundary detection and MIDI file creation. The team initially underestimated the difficulty of developing accurate boundary detection, resulting in a loss of time to work on other features. However, the team spent a few extra

days on the issue to meet the minimum requirement for a usable prototype, thus allowing the development of other features to continue. On the opposite side, MIDI file creation was estimated to take longer than it actually took to develop. The estimate was inaccurate due to lack of research and understanding into existing solutions available. In real world agile software development environments, task estimation is very difficult due to a lack of complete information and insight into related issues; accuracy only increases with experience.

6.0 Conclusion

Overall, the group created a strong prototype that represents the core functionality of the system original envisioned at the beginning of the fall term. A significant amount of research and development in the fields of signal processing and machine intelligence led to a robust classifier, and a tool that is a joy to use. The system withstood the unforgivingly noisy environment of the SYDE Design Symposium to an impressive degree. Users are able to train the system to associate distinct vocal sounds with specific audio outputs, and then the system can translate an input audio performance (such as a percussive beat-boxing performance or melodic humming performance) to an output file (of MIDI or WAV format) that accurately reflects the desired result. These outputs are very useful in the music production process, and reliably approximate the results envisioned by the user. As such, the system provides significant rapid music prototyping capabilities to aspiring producers, eliminating the intimidating requirements of time, money and skills that currently hinder the creative process. Essentially, this design project has made music production a lot less difficult.

Project MUSE has proven to be useful for aspiring music producers, with many expressing interest in using it for their own endeavours. Responses at demonstrations (both formal and informal) ranged from casual interest to outright enthusiasm from both professionals and amateurs alike. These reactions seem to confirm the market value for Project MUSE, and confirm the demand that was speculated to exist by the group at the beginning of the fall term. The project also shows great potential in the field of musical therapy, where it can be used to facilitate cognitive development and mood improvement in those with mental and physical disorders and disabilities.

Though the prototype was built to use a single type of input (vocal performance via microphone), its flexibility allows it to be expanded upon to include a variety of input formats as desired (such as table drumming, smartphone accelerometer, Microsoft Kinect, Leap Motion, and Thalmic Labs Myo). The flexibility of the MIDI standard allows the output to be used in a variety of applications as well (such as control over lighting, animation, motor control and video games). A variety of other improvements can be made in the future as well, including expansion of the test suite, increased

robustness of melodic performance interpretation and aforementioned expansion for new input and output formats.

Overall the project was quite successful as an initial prototype, and shows much future potential if pursued further. The group is grateful to all those who supported them along the way by lending their time and expertise, including their supervisor, professors, MBET students, and the people of FELT Lab.

References

[1] Vintage Synth Explorer. Akai MPC2000 / MPC2000 XL [Online]. Available: http://www.vintagesynth.com/akai/mpc2000.php

[2] Vintage Synth Explorer. Roland TR-808 Rhythm Composer [Online]. Available: http://www.vintagesynth.com/roland/808.php

[3] Billaboop. Boomclap is out! [Online]. Available: http://billaboop.com/en/boomclap

[4] M. Gao. (2012, October 12) Vocal Beater [Online]. Available: https://itunes.apple.com/us/app/vocal-beater/id384844110?mt=8

[5] Apple, Inc. Garage Band ’11 [Online]. Available: https://www.apple.com/ca/ilife/garageband/

[6] Apple, Inc. Logic Pro X [Online]. Available: https://www.apple.com/ca/logic-pro/

[7] Cakewalk. Sonar X3 [Online]. Available: http://www.cakewalk.com/products/sonar/

[8] Image-Line. FL Studio 11 [Online]. Available: http://www.image-line.com/documents/flstudio.html

[9] Amazon.com. Cakewalk: Software [Online]. Available: http://www.amazon.com/s/ref=bl_sr_software?_encoding=UTF8&field-brandtextbin=Cakewalk&node=229534

[10] Amazon.com. Akai: Musical Instruments [Online]. Available: http://www.amazon.com/s/ref=bl_sr_musical-instruments?_encoding=UTF8&field-brandtextbin=Akai&node=11091801

[11] Ableton Live 9. New in Live 9 [Online]. Available: https://www.ableton.com/en/live/new-in-9/

[12] D. Ellis. (2012, September 3). PLP and RASTA (and MFCC, and inversion) in Matlab using melfcc.m and invmelfcc.m [Online]. Available: http://labrosa.ee.columbia.edu/matlab/rastamat/

[13] D. Stowell, "Making music through real-time voice timbre analysis: machine learning and timbral control," Ph.D. thesis, School of Electronic Engineering and

Computer Science, Queen Mary University of London, London, Greater London, 2010.

[14] Leap Motion. (2014). SDK Documentation - Leap Motion 1.0.9 documentation [Online]. Available: https://developer.leapmotion.com/documentation/python/index.html

[15] OpenKinect. (2011, June 13). Python Wrapper [Online]. Available: http://openkinect.org/wiki/Python_Wrapper

[16] Thalmic Labs. (2014, February 20). Which programming languages could myo now be programmed in? dont overlook :) [Online]. Available: https://developer.thalmic.com/forums/topic/128/

[17] Wikipedia. (2013, August 4). MIDI usage and applications - Other applications of MIDI [Online]. Available: http://en.wikipedia.org/wiki/MIDI_usage_and_applications#Other_applications_of_MIDI

Appendix A

Script for wireframe testing

T: For the first task, please record a sound and link it to the snare drum in sound file

Q: How intuitive was this task? (1-5 with 1 being not intuitive and 5 being very intuitive)

Q: How many mistakes or miss-clicks did the user make?

Q: What was there any confusion in implementing this command? If so what was the confusion?

T: Please record a different sound and link it to the bass drum sound in sound file 2

Q: How intuitive was this task? (1-5 with 1 being not intuitive and 5 being very intuitive)

Q: How many mistakes or miss-clicks did the user make?

Q: What was there any confusion in implementing this command? If so what was the confusion?

T: Please create a beat using 2 snare drum sounds and 1 bass drum sound using your mouth

Q: How intuitive was this task? (1-5 with 1 being not intuitive and 5 being very intuitive)

Q: Would you use this application to create beats? (1-5 with 1 being never and 5 being always)

Q: How many mistakes or miss-clicks did the user make?

Q: What was there any confusion in implementing this command? If so what was the confusion?

Q: How would you improve the recording process?

T: Please play the sound clip back.

Q: How intuitive was this task? (1-5 with 1 being not intuitive and 5 being very intuitive)

Q: How many mistakes or miss-clicks did the user make?

Q: What was there any confusion in implementing this command? If so what was the confusion?

T: Please delete this sound clip.

Q: How intuitive was this task? (1-5 with 1 being not intuitive and 5 being very intuitive)

Q: How many mistakes or miss-clicks did the user make?

Q: What was there any confusion in implementing this command? If so what was the confusion?

T: Please create a new beat using 1 snare drum sound, 1 bass drum sound followed by 1 snare drum sound.

Q: How intuitive was this rerecording task? (1-5 with 1 being not intuitive and 5 being very intuitive)

T: Please export the file in WAV format

Q: How intuitive was this task? (1-5 with 1 being not intuitive and 5 being very intuitive)

Q: How many mistakes or miss-clicks did the user make?